SIAM J. Discrete Math.
Vol. 25, No. 3, pp. 1412–1417

PERMUTATIONS AS PRODUCT OF PARALLEL TRANSPOSITIONS*

CHASE ALBERT†, CHI-KWONG LI†, GILBERT STRANG‡, AND GEXIN YU†

Abstract. It was conjectured that a permutation matrix with bandwidth b can be written as a product of no more than $2b-1$ permutation matrices of bandwidth 1. In this note, two proofs are given to affirm the conjecture.

Key words. banded permutation, factorizaton, sorting network

AMS subject classifications. 05A05, 20B99, 15A23, 65T50

DOI. 10.1137/100807478

1. Introduction.

An $n \times n$ matrix $A = (a_{ij})$ is a b-banded matrix if $a_{ij} = 0$ whenever $|i-j| > b$. Thus, a 0-banded matrix is diagonal, and a 1-banded matrix is tridiagonal. In connection to factoring b-banded matrices with b-banded inverses, the following conjecture was posted in [4].

Conjecture 1.1. *Suppose P is a b-banded permutation matrix. Then P can be written as the product of no more than $2b-1$ 1-banded permutation matrices.*

We can formulate the conjecture in terms of a permutation as follows.

Conjecture 1.2. *Suppose $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is a permutation (bijection) with*

$$(\sigma(1), \ldots, \sigma(n)) = (i_1, \ldots, i_n)$$

such that all $|i_j - j| \le b$. Let $\mathcal{S}$ be the set of products of disjoint transpositions. Elements of S switch one or more disjoint consecutive pairs $(j, j+1)$. Then σ can be written as the product (composition) of no more than $2b-1$ elements in $\mathcal{S}$.

Evidently, the set $\mathcal{S}$ corresponds to the 1-banded permutation matrices. In this paper, we affirm the conjecture by giving two proofs. The first one depends on a greedy algorithm. The second proof uses sorting networks and the zero-one principle as in [1].

This conjecture deals with the special case of permutations in the main theorem of [4], [5]: If A and A^{-1} both have bandwidth b or less, then A can be factored into a product of block diagonal matrices of bandwidth 1. The number of factors of A depends on that number b and not on the matrix size n. (The matrix could be singly infinite, and this remains possible for our permutation matrices.) For permutations, where the inverse is the transpose, the only requirement is bandedness. An upper bound on the number of factors is not hard to establish. The conjecture states the best possible bound $2b-1$. At the end of this paper we describe the permutations that are most difficult to factor—the upper bound is attained.

*Received by the editors September 2, 2010; accepted for publication (in revised form) July 11, 2011; published electronically September 20, 2011.

http://www.siam.org/journals/sidma/25-3/80747.html

†Department of Mathematics, College of William & Mary, Williamsburg, VA 23185 (caalbe@email.wm.edu, ckli@math.wm.edu, gyu@wm.edu). The research of the second author was supported in part by NSF grant DMS-0914670, and the research of the fourth author was supported in part by NSF grant DMS-0852452.

‡Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139 (gs@math.mit.edu).

Panova [2] has found a particularly elegant proof based on wiring diagrams. Her approach shows all factors at once, where our construction reaches the identity permutation by a sequence of parallel exchanges of neighbors (which are permutations of bandwidth 1). This resembles the "bubblesort" algorithm, not a favorite in the analysis of sorting. By allowing parallel transpositions, the iteration count (less than $2b$) becomes respectable. A further proof has been proposed by Samson and Ezerman [3].

2. A greedy algorithm and proof of Conjecture 1.2.

Greedy Algorithm. Set $i_0 = 0$ and $i_{n+1} = n+1$. For each maximal subsequence $(i_r, i_{r+1}, \ldots, i_{r+s})$ with consecutive terms in the permutation, so that

$$i_{r-1} < i_r > i_{r+1} \ldots > i_{r+s} < i_{r+s+1},$$

interchange all the pairs i_{r+2t} and i_{r+2t+1} for $0 \le 2t < s$. Repeat this greedy step until the sequence is strictly increasing.

We will prove the following theorem, which implies the truth of Conjecture 1.2.

Theorem 2.1. *Suppose $(i_1, \ldots, i_n)$ is a permutation of $(1, \ldots, n)$. If $|i_j - j| \le b$ for all $j = 1, \ldots, n$, then in no more than $2b-1$ greedy steps, $(i_1, \ldots, i_n)$ will be restored to $(1, \ldots, n)$.*

This will hold true once we show the following.

Lemma 2.2. *If $|i_j - j| \le b$ for all $j = 1, \ldots, n$, then for every inversion (i_x, i_y) with $x < y$ and $i_x > i_y$, the number i_y will appear on the right side of i_x in no more than $2b-1$ greedy steps.*

Proof. To prove Lemma 2.2, let $i_\ell = k \in \{1, \ldots, n\}$. Assume that there are p elements larger than k lying on its left in the permutation $(i_1, \ldots, i_n)$, and $u_1 < \cdots < u_p < \ell$ satisfy $i_j > k$ for $j \in \{u_1, \ldots, u_p\}$. Also, assume that there are q elements smaller than k lying on its right in the permutation $(i_1, \ldots, i_n)$, and $\ell < v_1 < \cdots < v_q$ satisfy $i_j < k$ for $j \in \{v_1, \ldots, v_q\}$. All the pairs (i_{u_j}, k) and (k, i_{v_j}) constitute all the inversions that include k. The following claims can be verified readily.

Claim 1. The following statements are true:

(a) Suppose $|k - \ell| < b$. Then u_1 is minimized when $i_{u_1} = k+1$ and $u_1 = k+1-b$, and v_q is maximized when $i_{v_q} = k-1$ and $v_q = k-1+b$. Consequently, $v_q - u_1 \le 2b-2$.

(b) Suppose $\ell - k = b$. Then $q = 0$ and u_1 is minimized when $i_{u_1} = k+1$ and $u_1 = k+1-b$ so that $\ell - u_1 = 2b-1$, and in this extremal case,

$$(i_{u_1}, \ldots, i_\ell) = (k+1, \ldots, k+b, k-b+1, \ldots, k).$$

(c) Suppose $k - \ell = b$. Then $p = 0$ and v_q is maximized when $i_{v_q} = k-1$ and $v_q = k-1+b$ so that $v_q - \ell = 2b-1$, and in this extremal case,

$$(i_\ell, \ldots i_{v_q}) = (k, k+1, \ldots, k+b, k-b, k-b+1, \ldots, k-1).$$

Next, we turn to the proof of the following.

Claim 2. Suppose $k' = i_j$ for some $j \in \{u_1, \ldots, u_p\}$. In fewer than $2b-1$ greedy steps, k' will be moved to the right side of k.

Let us focus on the changes of positions of the terms in the set

$$S = \{i_z : u_1 \le z \le v_q\}$$

after a sequence of greedy steps is applied to $(i_1, \ldots, i_n)$.

If Claim 1(c) holds, there is nothing to prove. Assume Claim 1(b) holds. Then the subsequence $(i_{u_1}, \ldots, i_{v_q})$ has no interaction with other terms in $(1, \ldots, n)$. One readily verifies that the order of $(i_{u_1}, \ldots, i_{v_q})$ will be restored in $2b-1$ steps.

Assume that Claim 1(a) holds. Here are a few observations about the greedy step.

(1) The greedy step will never move a smaller number to the right side of a larger number.

(2) An element w will move to the right (switch position with its right neighbor) if $w = i_{r+2t}$ in a certain decreasing subsequence $(i_r, \ldots, i_{r+s})$ as described in the greedy step.

(3) In a greedy step, an element w will move to the left (switch position with a left neighbor) if $w = i_{r+2t+1}$ in a certain subsequence $(i_r, \ldots, i_{r+s})$ as described in the greedy step.

(4) In a greedy step, an element w will not move if the right neighbor is larger and one of the following holds.

(4.1) The left neighbor of w is smaller,

(4.2) $w = i_{r+s}$ in a certain subsequence $(i_r, \ldots, i_{r+s})$ as described in the greedy step, where s is even, so that the left neighbor of w is larger.

(5) If there are d numbers larger than k lying on the left side of k in the permutation $(\sigma(1), \ldots, \sigma(n))$, then in a sequence of greedy steps, k will move to the left at most d times.

(6) If $k' > k$ is such that k' lies on the left of k and there are r numbers larger than k' lying between k and k', then k' may remain stationary for $r+1$ greedy steps (not necessarily consecutive steps) while k is not moving left. This maximum delay occurs, for example, when the r elements lie immediately to the right of k'.

Proof. Note that a number does not move to the right if and only if its right neighbor is larger and one of (4.1) or (4.2) holds. (Note also that when (4.2) happens, either some larger element is right next to k', in which case we may regard the delay as caused by the numbers on the right, or some smaller number is right next to k', in which case k' will move to the right.) So we may only consider the delays caused by the numbers on the right of k'.

Let $k_1, k_2, \ldots, k_r$ be the r elements between k' and k. We may assume that there are enough small elements between k_r and k so that k does not move left when k' remains stationary. We observe that a small element will be moved from the right of k_r to the left of k_1 in every $r+1$ consecutive greedy steps (note that no small element may be moved in the first step). So k' will move in the $(r+2)$th step, and at that step, another smaller element will be right after k'. Therefore, k' will not remain stationary after $r+1$ steps until k starts to move left.

Note that $i_t < k$ if $t < u_1$ and $i_s > k$ if $s > v_q$. By observation (1), k cannot be moved to the left of u_1 nor to the right of v_q by the greedy steps.

Suppose $q > 0$. By observation (5), k may be moved to the right side by q greedy steps, in which k is most difficult for k' to overtake. We may assume that this worst case happens, and k is moved to the right by q steps before it is overtaken by k', so k will be at the $i_\ell + q$ position.

Assume that there are r elements on the left of k that are also larger than k', and among them, r_1 elements are on the left of k' so that $i_j - r_1 \geq u_1$. Then by observation (5), k' may be moved to the left by r_1 greedy steps before k' overtakes k. After that, all r elements may force k' to remain stationary for $r+1$ greedy steps, by observation (6).

On the other hand, those r elements will move to the right of k before k', so they will force k to move left for r positions before k' overtakes k. Therefore, k' may need to move

$$\begin{aligned}(i_\ell + q) - (i_j - r_1) - r &\le v_q - u_1 - r \\ &\le 2b - 2 - r \qquad \text{(by Claim 1(a))} \\ &= 2b - r - 2\end{aligned}$$

positions to overtake k. Since k' will move (to the right) in every greedy step other than the $r+1$ stationary ones, k' overtakes k in at most $2b-1$ greedy steps. ∎

3. Sorting network and another proof of Conjecture 1.2. A sorting network is a model of a network of wires and comparisons that is used to sort a sequence of numbers. Each comparison connects two wires and sorts the values on those wires by swapping them if they are out of order. In a graphical representation of comparison networks, one lines the numbers to be sorted vertically on the left side and the desired output $1, \ldots, n$ on the right side. Wires runs horizontally and comparisons are wired vertically.

For example, we consider the following odd-even algorithm and even-odd algorithm.

ODD-EVEN ALGORITHM.

Step 1. Compare (i_{2k-1}, i_{2k}) for all $k \ge 1$, and swap them whenever $i_{2k} < i_{2k-1}$.

Step 2. Compare (i_{2k}, i_{2k+1}) for all $k \ge 1$, and swap them whenever $i_{2k+1} < i_{2k}$.

Alternate these two steps until the sequence is strictly increasing.

Example: Apply the odd-even algorithm to the permutation (2, 3, 4, 1); we get the following steps: $(2,3,4,1) \to (2,3,1,4) \to (2,1,3,4) \to (1,2,3,4)$.

EVEN-ODD ALGORITHM.

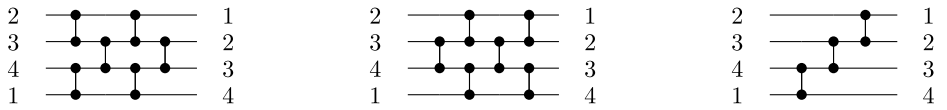
Step 1. Compare (i_{2k}, i_{2k+1}) for all $k \ge 1$, and swap them whenever $i_{2k+1} < i_{2k}$.

Step 2. Compare (i_{2k+1}, i_{2k+2}) for all $k \ge 1$, and swap them whenever $i_{2k+2} < i_{2k+1}$.

Alternate these two steps until the sequence is strictly increasing.

Example: Apply the even-odd algorithm to the permutation (2, 3, 4, 1); we get the following steps: $(2,3,4,1) \to (2,3,4,1) \to (2,3,1,4) \to (2,1,3,4) \to (1,2,3,4)$.

These algorithms give an odd-even network and an even-odd sorting network, respectively. Their graphical representations could be the full odd-even sorting network on the left, the full even-odd sorting network in the middle, and the minimal network on the right.



A path in a sorting network is a left-right route that possibly switches wires at the comparisons. We can usually use "depth" to evaluate how effective a sorting algorithm is. The depth is the maximum number of comparisons along any path from an input to an output. In other words, the depth is the minimum sorting time in a network which allows nonoverlapping comparison-exchanges at the same time. The above comparison networks are of depth 4, 4, and 3, respectively.

Our main theorem in this section is the following

THEOREM 3.1. *Each permutation π with bandwidth b can be sorted by a network with depth $2b-1$.*

It is clear that Conjecture 1.2 follows from Theorem 3.1.

We first express any permutation $(i_1, \ldots, i_n)$, using n binary sequences $f_1, f_2, \ldots, f_n$. The 0–1 sequence f_k has zero in position j if $i_j \le k$.

For example, the sequences f_2 and f_8 for the permutation $\pi = (4, 2, 1, 7, 5, 3, 9, 8, 6)$ are

$$f_2(\pi) = (\,1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\,),$$
$$f_8(\pi) = (\,0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\,).$$

It is clear that each permutation π uniquely corresponds to the n binary sequences $f_1(\pi), f_2(\pi), \ldots, f_n(\pi)$.

The followingis a well-known result of sorting networks.

THEOREM 3.2 (the zero-one principle; see [1]). *An n-line comparison network is a sorting network if and only if it sorts all binary sequences of length n.*

So to prove Theorem 3.1, it suffices to show that the network can sort all sequences $f_1(\pi), f_2(\pi), \ldots, f_n(\pi)$ for a permutation π with bandwidth b in $2b-1$ steps. Each binary sequence $f_k(\pi)$ has the form $(0, \ldots, 0, W_k, 1, \ldots, 1)$, where the "window" W_k is a binary subsequence *starting with a* 1 and *ending with a* 0.

LEMMA 3.3. *For a b-banded permutation π, if $f_k(\pi) = (0, \ldots, 0, W_k, 1, \ldots, 1)$, the window W_k has length at most $2b$.*

Proof. By the constraint of bandwidth, $f_k(i_j) = 0$ for $j < k - b$, and $f_k(i_j) = 1$ for $j \ge k + b$. So the first possible one in W_k is i_{k-b}, and the last possible zero is i_{k+b-1}. Thus the length of W_k is at most $2b$. ∎

LEMMA 3.4. *Every W_k can be sorted by an odd-even sorting network of depth at most $2b$. Furthermore, $W_k = (1, 1, \ldots, 1, 0, \ldots, 0)$ with b ones and b zeros is the only subsequence that needs depth $2b$.*

Proof. We only consider the case when W_k has length $2b$, since any other window is a subsequence of some window with length $2b$. Note that in W_k, at most b elements are at most k and at most b elements are more than k. So the output is $I_k = (0, \ldots, 0, 1, \ldots, 1)$ with b ones and b zeros. We count the maximum time (which consists of sorting time and delay time) in the minimal backward sorting network, that is, in how many steps we may get W_k from I_k.

Observe that once all zeros are in the correct places, the ones are too. So we just need to consider the movement of zeros. By the constraint of bandwidth, every zero needs to move at most b steps. In particular, the first zero in I_k needs to move at most b steps.

Now we consider the maximum delay time of the odd-even sorting network. Note that a delay can occur only when a zero is compared with another zero or is not properly aligned in the comparison due to parity of its position.

When b is odd, the bth zero is at odd position in W_k, so there is no delay for this zero in I_k. Thus the bth zero will move in every step. It follows that after the first step, the $(b-1)$th zero will move in every step. Furthermore, for any k with $1 \le k < b$, the delay time of the kth zero is $b - k$, caused by the $b - k$ zeros on its right. So the maximum delay time is $b - 1$.

When b is even, the bth zero will not move in the first step. After the first step, the process is the same as the case when b is odd. So the whole process is delayed once by the odd-even sorting network. Therefore the maximum delay time is at most b.

Therefore, the maximum total time is at most $b + b = 2b$. Note that when the maximum time equals $2b$, the first zero in I_k moves b steps, thus all other zeros need to move b steps as well. Thus $W_k = (1, 1, \ldots, 1, 0, \ldots, 0)$ with b zeros and b ones. ∎

Now we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. We call $(i_{m_1}, \ldots, i_{m_2})$ a *block* of π if

(1) for any $j < m_1$, $i_j < \min\{i_{m_1}, \ldots, i_{m_2}\}$, and for any $j > m_2$, $i_j > \max\{i_{m_1}, \ldots, i_{m_2}\}$;
(2) no subsegment of $(i_{m_1}, \ldots, i_{m_2})$ has property (1).

We first partition π into blocks. It is clear that no comparison will exchange elements of different blocks. So we may assume that π consists of a single block.

We now show that there is a sorting network with depth $2b-1$ that will sort π. Consider the binary sequences $f_1(\pi), \ldots, f_n(\pi)$. By Theorem 3.2, it suffices to show that there is a sorting network with depth $2b-1$ that will sort every $f_i(\pi)$.

By Lemma 3.4, the odd-even sorting network with depth $2b-1$ will sort all except one sequence (window). The special window $W_k = (1, 1, \ldots, 1, 0, \ldots, 0)$ with b zeros and b ones is a block. For this block, we choose an even-odd sorting network (note that the sorting in the block is independent of the rest of the permutation). Since there is no delay at the first step, it can be sorted in at most $b + (b-1) = 2b-1$ steps. ∎

4. Final remarks. Some stronger statements may be obtained, based on ideas from our second proof.

We still consider only permutations of bandwidth b and of a single block. For a given such permutation π, we consider the windows $f_1(\pi), f_2(\pi), \ldots, f_n(\pi)$. Since all the windows are binary sequences of length at most $2b$, we can further determine the steps it takes to sort *each* individual window using the even-odd sorting network or the odd-even sorting network. If the maximum depth required is d, then we can build a sorting network for the permutation with depth $d+1$, by extending the network of depth d to the whole permutation.

With some effort, we have been able to strengthen our conclusion to the following:

If a permutation does not contain the segment $(b+1, b+2, \ldots, 2b, 1, 2, \ldots, b)$ or $(b+1, b+2, \ldots, 2b-1, b, 1, 2, \ldots, b-1)$ or a shift of those segments as blocks, then it can be sorted in at most $2b-2$ steps.

REFERENCES

[1] D. E. Knuth, *Art of Computer Programming*, Vol. 3, *Sorting and Searching*, 2nd ed., Addison–Wesley, Reading, MA, 1998.

[2] G. Panova, *Factorization of banded permutations*, Proc. Amer. Math. Soc., to appear.

[3] M. D. Samson and M. F. Ezerman, *Factoring Permutation Matrices into a Product of Tridiagonal Matrices*, manuscript, http://arxiv.org/abs/1007.3467.

[4] G. Strang, *Fast transforms: Banded matrices with banded inverses*, Proc. Natl. Acad. Sci. U.S.A., 107 (2010), pp. 12413–12416.

[5] G. Strang, *Groups of banded matrices with banded inverses*, Proc. Amer. Math. Soc., 139 (2011), pp. 4255–4264.